SHUFFLE THOSE DIGITS

Our series of short entertaining programs continues with a look at a puzzle known as Reverse. The object of the game is to arrange a list of numbers in ascending order in the least number of moves. This may seem exceedingly simple, but often the best puzzles stem from the simplest concepts.

The program randomly generates a list of numbers for sorting. Changing the order of the numbers is possible only by reversing specified groups within the list. For example, if the computer generates the following random list in response to the player's request for nine numbers:

2 8 4 7 1 5 6 9 3

and the player then specifies 'Reverse? 5', the first five numbers will be inverted and the list becomes:

1 7 4 8 2 5 6 9 3

It shouldn't take you too much time and effort to solve a puzzle like this, and you would expect that it could be easily solved by a predefined algorithm. In practice, however, it is difficult to define a really good one. Suppose that there are n numbers in the list. The most obvious algorithm is this:

- Find the largest number in the list and reverse all the numbers up to its position. The largest number is now at the left-hand end of the list.
- Reverse all n numbers so that the largest number is in its desired position at the right-hand end of the list. This has taken only two reverses.
- Find the second largest number and repeat the whole procedure again. To move this number to its desired location requires a 'Reverse n-1' move.
- Repeat the procedure until the order is obtained.

This algorithm *always* solves the puzzle in 2n-3 moves. But it is possible to achieve a solution in fewer moves than this. To demonstrate how a forward-looking strategy can reduce the number of turns, consider the example we give in the box. Our algorithm would take seven (2×5-3) turns, but a skilled player could do it in four.

This program is a simple example of a whole series of reversing games that people have created and explored. You might like to try your hand at developing games that reverse from either end of the line, or where you have to sort out a grid of numbers rather than just a line. If you do design your own version of the game, you might like to jazz it up by using different coloured blocks to replace the numbers. The object of the game could then be to rearrange a line of blocks to match a pattern of coloured blocks at the top of the screen. You might also like to try incorporating an algorithm into the program to help a player who gets stuck.

Reverse

```
 20 DIM a(20)
 30 CLS : PRINT "Reverse !"
 40 INPUT "How many numbers ? ";n
 50 IF n<0 OR n>20 OR n<>INT n THEN  GO TO 30
 60 REM Jumble the list
 70 FOR i=1 TO n: LET a(i)=i: NEXT i
 80 RANDOMIZE
 90 FOR i=1 TO n
100 LET r=INT (RND*n+1)
110 LET x=a(r): LET a(r)=a(i): LET a(i)=x
120 NEXT i
130 LET t=1
135 REM Print the board
140 CLS : PRINT "Turn ";t;": The list is:": PRINT
150 FOR i=1 TO n: PRINT a(i);" ";: NEXT i
152 REM Check for a win
154 LET i=1
156 IF a(i)=i THEN  LET i=i+1: IF i<=n THEN  GO
TO 156
158 IF i>n THEN  GO TO 230
159 REM Get a go
160 PRINT : PRINT : INPUT "Reverse?";r
170 IF r<>INT r OR r<0 OR r>n THEN  GO TO 140
175 REM Reverse r
180 LET t=t+1
190 FOR i=1 TO INT (r/2)
200 LET x=a(i): LET a(i)=a(r-i+1): LET a(r-i+1)=x
210 NEXT i
220 GO TO 140
230 REM A Winner!
240 PRINT : PRINT : PRINT "You finished in ";t;"
turns"
250 PRINT : INPUT "Play again (y/n) ? ";a$
260 IF a$="Y" OR a$="y" THEN  RUN
270 CLS : STOP
```

Basic Flavours

On the Commodore 64 and Vic-20, replace RANDOMIZE by XX=RND(-TI), replace RND*N by RND(1)*N, and replace CLS by PRINT CHR$(147)

On the BBC Micro delete line 80, and replace R=INT(RND*N+1) by R=RND(N)

On the Oric-1 and the Oric Atmos, delete line 80, and replace RND*N by RND(1)*N

About Turn

The object of Reverse is to sort a list by repeatedly reversing subsets of it. The section to be reversed must always start with the leftmost element, and is described by the number of elements included. Here the successful sequence of moves is 2-3-5-3, meaning 'Reverse the leftmost two cards, then the leftmost three, etc.'